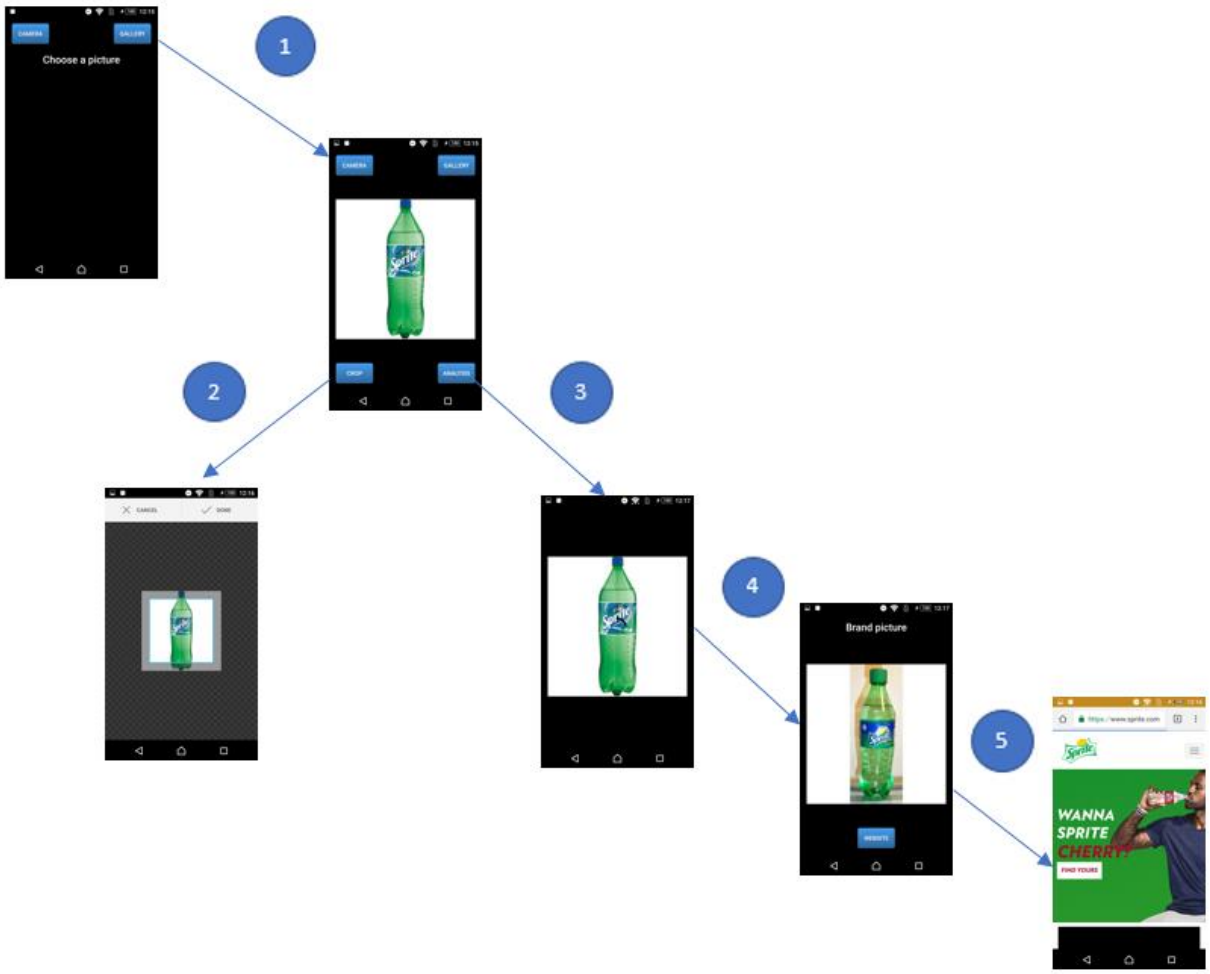
Projet PIIM

Présentation du projet

Ce projet nous a été donné par nos professeurs d'informatique, H. Wannous et G. Vanwormhoudt, dans le cadre du module A111-T : Projet Mobil Image. J'ai réalisé ce projet seule (projet à réaliser en binômes, mais nous sommes 11).

L'objectif de ce projet est d'offrir une application Android de détection de logos, en capturant une image (ou en en sélectionnant une existante), l'analysant, et trouvant la marque la plus représentative de l'image dans les marques disponibles.

Utilisation de l'application : Schéma fonctionnel



1 – Choix de l'image par l'appareil photo ou par la galerie.

2 – Possibilité de faire un crop sur l'image choisie. Le crop est à taille variable.

3 – Analyse : lorsqu'on appuie sur le bouton, tous les boutons disparaissent, et une animation d'attente apparaît.

4 – Quand l'analyse est terminée, l'image de référence de la marque apparaît.

5 – Quand on clique sur le bouton du site web, on ouvre une page web du site de la marque correspondant à l'image de référence., on ouvre une page web du site de la marque correspondant à l'image de référence.

Traitement des images

Ce projet s'est déroulé en deux étapes : la production d'une première et d'une seconde version de l'application, en fonction du traitement des images.

Version 1

Cette première version a deux caractéristiques principales liées au traitement de l'image. L'image devait être analysée grâce à des images de référence stockées en local, et utiliser une méthode de traitement particulière. En effet, afin de trouver la marque la plus représentative de l'image, il fallait détecter et comparer des points clés entre l'image sélectionnée et toutes les images de chaque marque. Si le nombre de ces points de comparaison était suffisamment bon, on devait alors trouver l'image qui avait les meilleurs points de comparaison avec celle sélectionnée.

Version 2

Cette deuxième version a également deux caractéristiques principales, différentes de la première version. Tout d'abord, les images de référence ne se trouvent plus en local, mais sur un serveur distant. Ensuite, la méthode de traitement de l'image est différente. À partir de ces images de référence, pour chaque marque, un sac de mots (Bag of Words) est réalisé, pour former un dictionnaire, retranscrit sous la forme d'un fichier yml (vocabulary.yml). Chaque marque possède à présent son propre classifier (nomMarque.xml), à partir duquel un histogramme de comparaison avec l'image sélectionnée est créé. Le meilleur histogramme permet alors de trouver la classe la plus représentative de l'image sélectionnée. Les classifiers et le vocabulaire se trouvent sur le serveur, nous n'avons pas eu à les créer.

Environnement logiciel utilisé

Afin de mener à bien ce projet, nous avons utilisé les logiciels suivants :

- Android Studio comme environnement de développement,
- OpenCV et JavaCV pour le traitement des images,
- Java sous Android comme langage,
- Git et GitHub comme gestionnaires de configuration,
- Les librairies Volley pour les requêtes vers le serveur,
- La librairie android-crop pour la méthode de crop.

J'ai également utilisé un smartphone Sony Xperia Z2 et un Huawei P9 pour tester mon application.

Utilisation des classes

Mon application possède cinq classes, dont voici les caractéristiques :

ChoosePic

Cette classe gère l'activité de départ de l'application. Toute la gestion de capture de l'image (par caméra et galerie) s'y fait. Ainsi, on va avoir des méthodes de :

- Gestion des permissions pour accéder à la galerie et la mémoire du téléphone (lecture et écriture).
- Création de fichier à partir de la capture d'une photo.
- Redimensionnement d'image.
- Récupération d'URI à partir d'un bitmap.
- Crop d'une image.

C'est également de là que partent les appels aux autres classes et à l'activité de résultat. On y trouve aussi un thread dans lequel l'analyse de l'image s'effectue, et pendant laquelle une barre de progression indéterminée (animation d'un cercle d'attente) s'affiche tant que le traitement n'est pas terminé. Un jeu sur la visibilité des boutons permet de s'assurer que l'analyse et le crop ne peuvent se faire qu'une fois l'image choisie, et que lors de l'analyse aucun bouton ne soit touché.

La gestion des boutons dans cette classe est un peu particulière. En effet, j'utilise deux manières de lancer des activités sur les boutons. Dans le cas de la caméra et de la galerie, j'utilise cette manière dans la méthode onCreate :

```
bCamera.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        dispatchTakePictureIntent();
    }
});
```

Alors que pour les boutons de crop et d'analyse, je me sers de la méthode onClick (en dehors de la méthode onCreate), avec un switch pour choisir le bouton :

```
//Button to crop picture
case R.id.bCrop:
    imageView.setImageDrawable(null);
    uriPic = Uri.parse("file:///"+mCurrentPhotoPath);
    Log.d("Crop-URI", uriPic.toString());
    beginCrop(uriPic);
    break;
```

J'ai décidé de conserver ces deux manières de faire, afin de garder un exemple de chaque. Je ne suis pas certaine de la meilleure méthode à utiliser, elles me semblent toutes les deux équivalentes, et comme pour moi les boutons d'analyse et de crop sont à utiliser après les boutons de caméra et de galerie, j'ai pensé qu'il était acceptable de garder les deux façons de faire, pour plus de lisibilité.

Requests

Cette classe permet de gérer la requête de l'application vers le serveur pour récupérer le fichier json contenant les marques (nom, site web, nom de classifier, nom de l'image de référence), et le nom du fichier de vocabulaire (utilisé plus tard dans le traitement de l'image). Le fichier est parsé et les informations recueillies vont remplir une liste d'objets de la classe Brand.

De temps en temps, le serveur met un peu de temps à répondre, et les fichiers ne sont pas chargés immédiatement, donc il est parfois (rarement) nécessaire d'attendre une à deux secondes pour appuyer sur le bouton Analysis, qui lance les requêtes vers le serveur.

Brands

Cette classe récupère les attributs de chaque marque comme mentionné plus haut, et possède des méthodes qui permettent, à partir de ces informations, d'aller récupérer les fichiers de classifier et d'image de référence par marque. Ainsi, lorsqu'une instance de la classe Requests crée autant d'objets de type Brands que le fichier json lui indique, deux méthodes de Brands vont récupérer le classifier et l'image de référence pour chacune de ces marques, à partir du nom du classifier et du nom de la première image de référence. À terme, il sera facile d'aller chercher d'autres images de la même manière, s'il est nécessaire d'en afficher plusieurs (voir difficultés spécifiques, Sérialisation et Parcelisation, pour comprendre pourquoi ce n'est pas réalisé actuellement).

PicAnalysis

Cette classe permet d'analyser l'image choisie et, avec un dictionnaire et des classifiers récupérés sur le serveur, de trouver par le système de Bag of Words, quelle marque est la plus représentative de la photo sélectionnée. Cette classe a donc trois méthodes principales :

- Une méthode pour récupérer le dictionnaire (vocabulary.yml), commun à toutes les marques.
- Une méthode pour récupérer les classfiers de toutes les marques.
- Une méthode qui trouve la marque la plus représentative de l'image sélectionnée (voir partie traitement de l'image).

ResultAnalysis

Cette classe gère l'activité en charge de l'affichage de l'image de référence de la marque et du bouton qui mène au site web de ladite marque. Elle récupère l'intent de l'activité précédente, qui contient le bundle avec l'Uri et le nom de la marque.

Difficultés rencontrées

Lors de la réalisation du projet, j'ai rencontré un certain nombre de difficultés.

Globales

Manque de compétences

N'ayant jamais développé sous Android, je n'ai pas forcément su comprendre les erreurs que j'ai rencontrées, surtout que les messages d'erreur remontés n'étaient pas toujours très parlants, même en recherchant sur le web des cas similaires. Je n'ai pas compris du coup pourquoi deux codes identiques ne fonctionnaient pas de la même manière, et c'est sur ce point que j'ai perdu le plus clair de mon temps. Par exemple, j'ai copié collé des morceaux de code donnés par nos professeurs, sans me rendre compte que certaines parties n'étaient pas copiables : l'environnement de développement étant différent, certaines subtilités m'ont échappé très longtemps, et je ne comprenais pas ce que mon professeur attendait de moi (ce qui a mené à quelques soucis de communication). Ce n'est qu'en copiant des parties beaucoup plus petites de code que j'ai pu remarquer ces subtilités, et en apprendre plus sur le fonctionnement d'Android.

Absence de documentation

La librairie d'OpenCV et de JavaCV nous a causé beaucoup de problèmes, à tous. En effet, même si nous comprenions le principe de traitement de l'image, il nous a été très difficile de nous approprier les méthodes de ces librairies, notamment à cause du grand manque de documentation. Internet ne nous a pas permis de trouver suffisamment de documentation et d'exemples pour nous aider à manipuler cette librairie sous Android. Ceci a mené à beaucoup de frustration, car nous n'avions pas eu ces difficultés lors de nos tentatives en Java (sous IntelliJ par exemple, hors projet Android).

Spécifiques

Sérialisation et Parcelisation

Lors du passage de l'intent de l'activité ChoosePic vers l'activité ResultAnalysis, je n'ai pas pu passer l'objet Brands complet. En effet, les bitmaps et les Uri ne sont pas sérialisables. J'ai tenté de parceler l'objet, sans succès, car je ne maîtrise pas du tout cet aspect. J'ai donc opté pour la création d'un Bundle qui récupère le nom de la marque et l'Uri de l'image de référence pour les mettre dans l'intent. Cependant, cette méthode est fastidieuse, et si j'avais dû afficher plusieurs images de référence, il aurait fallu que j'écrive, image par image, le code qui permet de mettre l'Uri de chaque image dans le bundle. Cette difficulté serait apparue lors de la version 3 du projet, qui a été abandonnée, donc je n'ai pas creusé la possibilité.

Images trop grandes

Une difficulté importante est venue du fait que les smartphones que j'ai utilisés pour tester l'application avaient une résolution d'image particulièrement élevée lors de la capture. Cela a causé parfois des erreurs de mémoire pleine (résolues en redimensionnant l'image capturée), et d'autres fois des erreurs aléatoire de fin d'application, à cause de la lenteur du traitement de l'image. Ce dernier type d'erreur est résolu avec un crop de l'image.

Bilan

Ce projet fut très difficile, mais également très riche. Difficile pour la première partie, principalement, où l'apprentissage d'Android avec OpenCV fut douloureux, et a causé des délais dans la restitution de cette partie. Mais la richesse que ce projet m'a apportée est indéniable : j'ai pu découvrir comment traiter des images, de diverses manières, ce qui me donne une vague idée du fonctionnement des moteurs de recherche (j'imagine que les méthodes sont beaucoup plus poussées et plus fines, mais il n'en reste pas moins que je peux à présent en comprendre le fonctionnement général). J'ai également appris à développer sous Android, ce qui me sera très utile par la suite, car le développement d'applications mobile est à la mode dans notre métier. Je sais à présent construire une application basique, jouer un peu avec les différentes activités dans leur fond et dans leur forme (styles, drawables, etc.).

Je regrette un peu de ne pas avoir pu travailler sur la troisième version du projet (annulée à cause du temps perdu sur nos difficultés), car j'aurais aimé apprendre à travailler sur l'indexation de bases de données (je ne l'ai jamais fait), et créer des services REST comme nous l'avons vu en cours. Néanmoins, ce projet est une belle entrée dans le monde du traitement de l'image et d'Android